UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

CRIAÇÃO DO MÓDULO CRM (SIGCRM), CRIAÇÃO DO MÓDULO DE FACTURAÇÃO ELECTRÓNICA NA APLICAÇÃO SIGGC E DESENHO DO KNOWLEDGE BASE

Rui Pedro Ribeiro Rodrigues

Mestrado em Engenharia Informática

2007

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

CRIAÇÃO DO MÓDULO CRM (SIGCRM), CRIAÇÃO DO MÓDULO DE FACTURAÇÃO ELECTRÓNICA NA APLICAÇÃO SIGGC E DESENHO DO KNOWLEDGE BASE

Rui Pedro Ribeiro Rodrigues

Projecto orientado pela Prof. Dr. Isabel Nunes e co-orientado por Fernando Amaral.

Mestrado em Engenharia Informática

2007

Declaração

Rui Pedro Ribeiro Rodrigues, aluno nº 28356 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "Criação do módulo CRM (SigCRM), criação do módulo de Facturação Electrónica na aplicação SigGC e desenho do Repositório de Conhecimento", realizado no ano lectivo de 2006/2007 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, 09 de Novembro de 2007

_____

Fernando Amaral, supervisor do projecto de Rui Pedro Ribeiro Rodrigues, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório de Projecto em Engenharia Informática, intitulado "Criação do módulo CRM (SigCRM), criação do módulo de Facturação Electrónica na aplicação SigGC e desenho do Repositório de Conhecimento".

Miraflores, 09 de Novembro de 2007

_____

Resumo

Este documento descreve os projectos realizados no âmbito da disciplina Projecto em Engenharia Informática do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa.

Os projectos decorreram nas instalações da Capgemini Portugal; o estágio teve a duração de nove meses.

O primeiro projecto consistiu na elaboração do módulo de facturação electrónica para a aplicação de gestão comercial do ERP da empresa, a SigGC. O módulo divide-se entre o processo de importação de documentos através de um serviço e o processo de exportação através de um mapa especial da aplicação SigGC.

O segundo projecto, intitulado *Knowledge Base*, consistiu em conceber um repositório de conhecimento recorrendo à tecnologia disponibilizada pela ferramenta *Sharepoint Services* da *Microsoft*. A sua principal funcionalidade é o agrupamento de informação por sectores categorizáveis

O terceiro projecto consistiu na implementação de uma aplicação *web* de CRM com a capacidade de gerir toda a informação relativa aos clientes de uma empresa, incluindo contactos, oportunidades relacionadas, contratos, preferências e relatórios de visita.

PALAVRAS-CHAVE:

Facturação Electrónica, *Knowledge Base*, *CRM*, *XML*, *ASP.NET*, *C#*, *VB6*, *Web*

Abstract

This document describes the work carried through in the scope of the class Projecto em Engenharia Informática (PEI) of the Mestrado em Engenharia Informática (MEI) in Faculdade de Ciências, Universidade de Lisboa.

All the work took place at the Capgemini Portugal installations, during nine months.

The first project consisted in an electronic billing module to integrate in SIG's commercial management application. This project is divided in two components, the import component, based on a Windows service, and the export component, implemented on a *Microsoft Excel* spreadsheet with background *VBA* support.

The second project had the goal of designing a knowledge base based on *Microsoft's Sharepoint Services* technology. Its main purpose is grouping information by different areas of interest.

Finally, last project's work was to develop a CRM web application capable of managing all the client related information.

KEYWORDS

Electronic Billing, Knowledge Base, CRM, XML, ASP.NET, C#, VB6, Web

Índice

Lista de Figuras

Capítulo 1

Introdução

Este relatório apresenta o trabalho realizado nos projectos Facturação Electrónica, *Knowledge Base* e SigCRM na empresa Capgemini, enquadrado na disciplina Projecto de Engenharia Informática no âmbito do Mestrado em Engenharia Informática do ano lectivo 2006/2007. O trabalho desenvolvido teve a duração de nove meses.

1.1 Motivação

Partindo do princípio que um cliente é considerado um bem imprescindível a qualquer empresa, surge a necessidade de obter o maior volume de informação possível sobre este. Obter informação acerca de um cliente significa conhecer os seus gostos, as suas necessidades, o seu historial e as suas motivações. Um conhecimento a este nível permite uma aproximação entre cliente e fornecedor e consequentemente a melhoria na qualidade da resposta às suas necessidades. Com o nível de globalização actual as fronteiras dos mercados diluem-se, a oferta e procura tornam-se diversificadas e sofrem mutações constantes. Assim, é de extrema importância marcar a diferença com ofertas personalizadas baseadas nas necessidades específicas de cada cliente.

Orientar a empresa para o cliente torna-se então um objectivo que uma aplicação como o *CRM* torna possível.

Do ponto de vista da entidade que utiliza o *CRM* as vantagens são também notáveis, entre elas a possibilidade de reunir, de forma centralizada e estruturada, a informação dos seus clientes, o que eleva a eficiência da gestão da carteira de clientes e permite um grau de controlo maior sobre o negócio.

Do conjunto de aplicações *CRM* existentes no mercado o SigCRM diferencia-se pelo facto de ser uma aplicação *web*. Seguem-se algumas das vantagens alcançadas através de uma aplicação com esta arquitectura:

- Instalação única (no servidor) para distribuir uma nova versão da aplicação;
- O acesso à aplicação torna-se independente da localização do utilizador;
- A aplicação e os dados não estão com o utilizador;

- O utilizador não tem de se preocupar com requisitos de *hardware*;

A motivação inerente ao projecto da Facturação Electrónica advém da necessidade de simplificação e dinamização do processo de comércio. Os seus benefícios aplicam-se tanto às empresas como ao próprio estado, contribuindo ainda para a sua modernização.

O projecto *Knowledge Base* surgiu com o intuito de fornecer uma ponte de comunicação entre a equipa de desenvolvimento e a equipa de assistência técnica do departamento em que estou inserido. O resultado permitirá reunir de forma centralizada a informação relativa aos problemas reportados pelos clientes, assim como a documentação de problemas já solucionados.

1.2 Breve descrição dos projectos

Esta secção é dedicada à descrição dos projectos realizados durante o estágio. Na secção 2.1 do capítulo 2 podem ser consultados os objectivos em detalhe de cada projecto.

O objectivo do projecto SigCRM é desenvolver um módulo de *CRM* para ambiente *web*, capaz de gerir os contactos e o relacionamento entre a organização que o utiliza e os seus clientes, de modo a que ao longo do tempo seja construído um historial das interacções, possibilitando uma melhor aproximação aos interesses e necessidades do cliente. A aplicação terá também a capacidade de registar todas as actividades da organização, como por exemplo, a agenda de planeamento geral, onde será possível consultar a actividade de cada colaborador num dado período de tempo. A deslocação de recursos humanos em serviço tende a ser uma actividade com pouco controlo sobre o trabalho realizado surgindo assim a necessidade de criar um arquivo de relatórios de visita. Outro objectivo do módulo é oferecer a possibilidade de gestão de processos, visto que a actividade de uma organização raramente se baseia apenas em tarefas singulares, mas sim em processos formados por um conjunto de tarefas – o *CRM* terá também a capacidade de gerir estes conjuntos de tarefas. A aplicação tem também como objectivo a capacidade de gestão de vendas, na vertente das oportunidades permitindo reunir informação sobre os potenciais clientes e o acompanhamento do processo de pré-venda e na vertente dos objectivos permitindo a análise das oportunidades concretizadas face às oportunidades identificadas. No campo da análise, os objectivos do *CRM* focam-se na possibilidade de fornecer dados sobre *performance* e estatística dos utilizadores, assim como do negócio em si. Através da aplicação será possível integrar as tarefas

criadas na aplicação de correio electrónico da *Microsoft* (o *Outlook*) e estará disponível a possibilidade da configuração do serviço de alertas através do envio e recepção de correio electrónico.

O objectivo do projecto da Facturação Electrónica é fornecer uma interface aos utilizadores da aplicação SigGC, que permita a importação e exportação de documentos em formato electrónico. Este projecto é desenvolvido em parceria com outra empresa, responsável pela aplicação que gere as comunicações de documentos electrónicos entre duas entidades e pelo repositório de armazenamento desses mesmos documentos.

No projecto *Knowledge Base* o principal objectivo é fornecer um serviço de repositório de informação, de fácil utilização, para centralizar e agilizar o processo interno de comunicação entre a equipa de desenvolvimento e de assistência do departamento.

1.3 Enquadramento institucional

Nesta secção descrevem-se as três disciplinas que compõem o corpo de negócio da instituição de acolhimento e descreve-se brevemente o ramo de cada uma delas. No final descreve-se a equipa de Aplicações *Standard* já que é nesta que fui inserido para realizar o estágio.

A Capgemini é uma das maiores companhias mundiais de Consultoria, Tecnologia e *Outsourcing*, trabalhando com os seus clientes de uma forma única e inovadora chamada *Collaborative Business Experience*. Pelo compromisso em atingir sucesso mútuo e a realização de valor tangível, a consultora ajuda as empresas a implementar estratégias de crescimento, potenciar a tecnologia, e prosperar através do poder da colaboração.

Os seus serviços estão organizados em três disciplinas:

I. ***Consulting Services:*** serviços na área de consultoria

Abrange as seguintes áreas:

- Estratégia e *Marketing*;
- Fusões e Aquisições;

- Optimização organizacional e de Recursos Humanos;
- Optimização funcional, incluindo Compras e TI;
- Gestão de Projectos e Aplicações;
- Design/Reestruturação de Modelos de Negócio e Governo;
- Serviços Partilhados;
- Gestão Comercial incluindo rentabilização de lucros do cliente;
- Modelos de Controlo de Gestão e *Balanced Scorecard*;

II. ***Technology Services:*** serviços na área das Tecnologias de Informação;

A empresa fornece serviços de arquitectura, integração e infra-estrutura baseados em soluções de TI. As suas competências em Tecnologias de Informação abrangem desde a avaliação ao planeamento, redefinição de processos, design, integração, implementação, migração e projectos por medida.

Esta disciplina cobre as seguintes áreas:

- Desenvolvimento e integração de aplicações;
- Tecnologia de agentes;
- Transformação de TI: Estratégia e Arquitectura;
- *Business Intelligence*;
- Infra-estrutura e Segurança;
- *Mobility Transformation*;
- *Open Source*
- Soluções Oracle;
- Portais;
- *Radio Frequency Identification*;
- *SAP*;
- *Siebel*;

III. ***Outsourcing Services:*** sector de serviços;

A empresa conta com resultados comprovados na obtenção da melhoria dos níveis de serviço, ganhos de eficiência e significativas reduções de custos, através da implementação das suas metodologias nos serviços de Outsourcing, que abrangem:

- Gestão de Aplicações;
- *Business Process Outsourcing*;
- Gestão de Infra-estrutura;
- *Transformational Outsourcing*;

Equipa de Aplicações *Standard* – SIG (Sistema Integrado de Gestão)

A equipa de Aplicações *Standard* é um grupo de trabalho de 35 elementos, divididos em três grupos funcionais distintos: a equipa de *help desk*, a equipa de comerciais e a equipa de desenvolvimento.

É através da equipa de Aplicações *Standard* da disciplina de *Outsourcing Services* que é desenvolvido o SIG. Este *software* é um *ERP* cuja função é armazenar, processar e organizar as informações geradas nos processos organizacionais agregando e estabelecendo relações de informação entre todas as áreas de uma empresa.

Um *ERP* é em termos gerais, uma plataforma de *software* desenvolvida para integrar os diversos departamentos de uma empresa, possibilitando a automatização e armazenamento de todas as informações de negócios.

Os módulos existentes actualmente para a plataforma *Windows* são:

- **SigGC**: Módulo de gestão comercial constituído por módulos (Administração, Vendas, Compras, Logística, Produção e Tesouraria);
- **SigCTB**: Módulo de contabilidade;
- **SigGP**: Módulo de gestão de recursos humanos;
- **SigAV360**: Sistema de avaliação de desempenho;

É neste contexto que serão desenvolvidos os projectos. O SigCRM é um módulo totalmente novo a ser adicionado aos já existentes enquanto o projecto Facturação Electrónica será incorporado no módulo SigGC. Finalmente o projecto *Knowledge Base* é uma ferramenta desenvolvida para uso interno da equipa da equipa das Aplicações *Standard*.

1.4 Organização do documento

Esta secção é dedicada à organização do documento que é composto por quatro capítulos, sendo este o primeiro intitulado introdução.

No segundo capítulo constam os objectivos dos projectos na secção 2.1, na secção 2.2 é explicada a metodologia utilizada e a terminar, na secção 2.3 está a informação sobre o planeamento.

No terceiro capítulo aborda-se a informação sobre as tecnologias utilizadas e o trabalho realizado nos três projectos – a secção 3.2.1 é dedicada ao projecto Facturação Electrónica, a secção 3.2.2 ao projecto *Knowledge Base* e a secção 3.2.3 ao projecto SigCRM.

No quarto, e último capítulo deste documento, intitulado Conclusão, constam o sumário e crítica do trabalho realizado na secção 4.1 e o trabalho futuro na secção seguinte.

Capítulo 2

Objectivos, metodologia e planeamento de trabalho

Neste capítulo do relatório descrevem-se pormenorizadamente os objectivos dos projectos na secção 2.1. A secção 2.2 é dedicada à descrição da metodologia seguida no planeamento, desenvolvimento e implementação dos projectos. Finalmente na secção 2.3 consta o planeamento do trabalho para os projectos.

2.1 Objectivos

Como já foi referido anteriormente, na secção 1.1 do primeiro capítulo, os clientes são o bem imprescindível de qualquer empresa assim, o módulo SigCRM terá de concretizar as funcionalidades que considerámos fulcrais para satisfazer os seus pontos de motivação. Os objectivos para a concretização destas funcionalidades passam então por desenhar uma aplicação correcta e flexível que reúna as condições certas para facilitar o desenvolvimento e a sua manutenção. Outro objectivo é a criação de uma interface gráfica intuitiva para interacção entre o modelo de dados criado e a aplicação. Ao nível de funcionalidades da aplicação os objectivos são a gestão do relacionamento com os contactos da empresa, sejam eles clientes, fornecedores ou outros, a possibilidade de gestão da força de vendas através do registo de oportunidades identificadas, a gestão de actividades, a criação e manutenção de relatórios de visita, a criação de uma agenda dos utilizadores e sincronização do planeamento pessoal com a aplicação de gestão de correio electrónico, *Microsoft Outlook,* a capacidade de efectuar análises de *performance* e estatísticas, gestão de despesas da empresa, registo de contactos recebidos/efectuados, e gestão de *mailing list*.

Os objectivos do projecto Facturação Electrónica são a criação de uma camada de abstracção necessária à transacção de documentos da aplicação SigGC para o serviço responsável pelo envio dos documentos em formato electrónico. Criar o mecanismo que irá efectuar a importação e exportação dos documentos. A exportação de documentos é efectuada através de um mapa especial que será criado e que terá a capacidade de exportar documentos da aplicação SigGC para qualquer um dos formatos electrónicos facultados. O processo de importação de documentos será conduzido por um serviço

Windows desenvolvido para o efeito. Após estes mecanismos terem sido desenvolvidos entra-se na fase final do projecto que passa pela implementação no cliente.

O *Knowledge Base* será criado com o propósito de fornecer um conjunto de informação dividida por sectores.

Um desses sectores diz respeito às ocorrências relacionadas com as aplicações *standard*. Neste, o objectivo é fornecer acesso rápido e eficiente a uma lista de ocorrências existentes, assim como simplificar a ligação entre o surgimento de uma nova ocorrência e o processo que se desencadeia para a sua resolução. O outro sector é a base de dados de conhecimento, que é um conjunto de documentos que têm a finalidade, entre outras, de descrever procedimentos, fornecer manuais de utilização e respostas a perguntas frequentes.

As principais funções fornecidas por este sistema serão a agregação de ocorrências num conjunto categorizável, de pesquisa simplificada e a possibilidade de consulta a documentos de resolução de ocorrências mantendo uma base de conhecimento com a finalidade das ocorrências futuras verem o seu processo de resolução simplificado e o seu tempo reduzido.

2.2 Metodologia

Durante o desenvolvimento do projecto, foi utilizada a metodologia *SCRUM*. Esta metodologia baseia-se nos conceitos descritos em seguida.

Cada projecto tem associada uma pessoa responsável, tipicamente denominada de *proprietária* do projecto. As suas funções passam por atribuir prioridades às tarefas que compõem o projecto e definir quais as tarefas que devem ser incluídas num determinado ciclo de desenvolvimento do projecto.

A *lista das tarefas* que compõem o projecto é outro dos conceitos desta metodologia. A sua organização por prioridades surge da necessidade de compreender, num dado ponto no tempo, quais as actividades que são realmente importantes para o desenvolvimento do projecto. Esta lista tem um carácter dinâmico, porque a sua criação coincide com o primeiro planeamento do projecto e nesta altura é impossível ter uma noção total e final de todas as tarefas necessárias à conclusão do mesmo.

Descritos os conceitos de proprietário do projecto e lista de tarefas, temos a definição do ciclo de realização de tarefas, denominado de *sprint*. Um *sprint* é um período de tempo em que a equipa realiza as tarefas a que se propôs. As tarefas de um *sprint* são definidas durante a reunião de planeamento. Nesta reunião os membros da equipa em conjunto com o proprietário do projecto devem chegar a acordo sobre quais as tarefas que podem ser realizadas a tempo do final do *sprint*. Os tempos de realização são unicamente estimados pela equipa, sendo função do proprietário do projecto tentar influenciar a equipa a realizar as tarefas que sejam deixadas de fora do *sprint* e que este considera importantes, por exemplo, reduzindo o âmbito de uma funcionalidade de modo a diminuir a estimativa da sua conclusão, dando lugar à realização de outra tarefa.

Durante um ciclo de desenvolvimento são feitas reuniões diárias (*scrums*) durante as quais é realizado o ponto de situação de cada elemento da equipa e onde são discutidas as dificuldades de cada elemento, o trabalho realizado no dia anterior e as expectativas para o trabalho nesse dia. Nestas reuniões existe uma pessoa responsável pela sua moderação – o *scrum master* além da função de moderação, este elemento tem também a incumbência de impedir que a equipa se comprometa durante a reunião de planeamento a realizar mais tarefas do que aquelas para as quais na realidade tem capacidade.

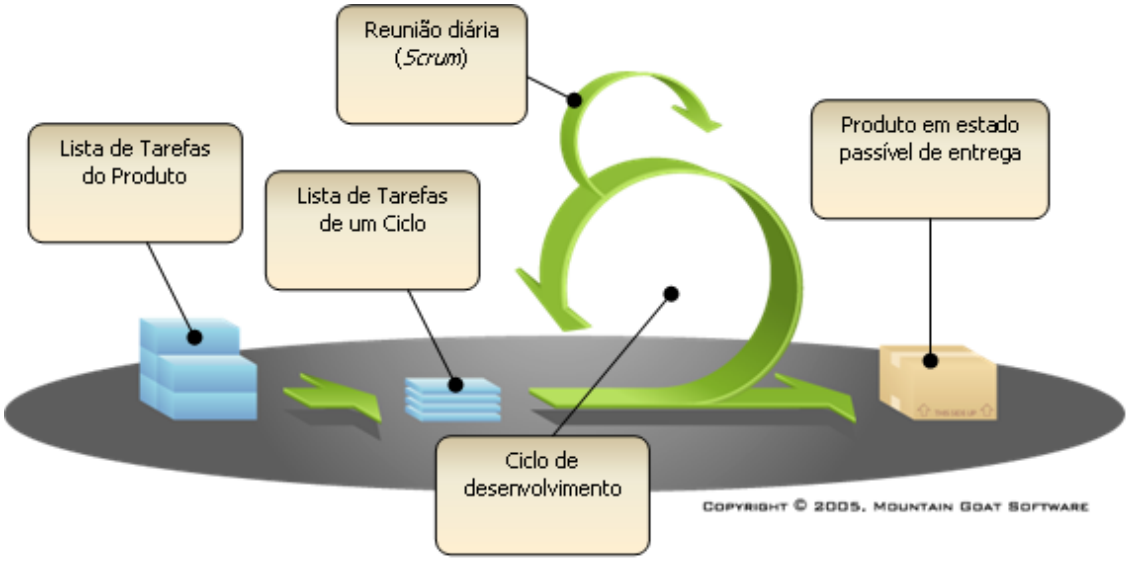

Figura 1 Esquema da metodologia SCRUM.

A equipa de desenvolvimento é composta por 18 membros – 5 seniores (gestores de projecto) e 13 juniores. Dentro da equipa de desenvolvimento existem sub-equipas criadas para projectos específicos.

2.3 Planeamento

O planeamento do trabalho a realizar dividiu-se em três fases. Na Tabela 1 pode ser consultado o esquema das actividades envolvidas em cada uma das fases.

A primeira fase consistiu no processo de formação e integração na empresa. Durante esta fase foram realizadas acções de formação interna, «*on the job*», de duração variável, preparadas e apresentadas por colegas, sobre as tecnologias utilizadas.

Na segunda fase deste estágio deu-se início aos projectos Facturação Electrónica e *Knowledge Base*. A realização destes dois projectos estendeu-se por quatro meses, sensivelmente dois meses para cada um deles. O projecto Facturação Electrónica foi desenvolvido por uma equipa de dois elementos – eu, como elemento júnior e o colega Paulo Figueiredo, como elemento sénior. Este projecto teve uma fase inicial de análise, efectuada pelo membro sénior em que foram reunidas as condições para início do desenvolvimento. Durante a fase de desenvolvimento, realizada totalmente por mim, foram criadas as interfaces para os vários formatos proprietários dos clientes e

desenvolvida a interface para integração no módulo de gestão comercial do SIG. Finalmente na última fase, tiveram lugar os testes e implementação do projecto.

O projecto *Knowledge Base*, contou também com as mesmas fases do projecto descrito anteriormente. A equipa envolvida neste projecto contou com 3 elementos – eu, como elemento júnior e os colegas Paulo Figueiredo e Sérgio Sarabando como elementos seniores. Na fase de análise, realizada pelos elementos seniores, foram definidas as necessidades conceptuais do repositório. Na fase de execução, que realizei na totalidade, foi desenhado o repositório e implementados os conceitos definidos na fase de análise. Na última fase o projecto foi testado, mas ainda não se encontra em produção.

Finalmente, a terceira fase foi a participação no projecto SigCRM que decorreu durante cinco meses e cuja equipa foi composta por 5 pessoas – o colega Daniel Vinagre, como gestor de projecto, eu e os colegas Sílvia Inácio, Miguel Marques e Hélder Marques como elementos juniores. A minha integração no projecto aconteceu quando este já estava na fase de implementação.

Primeira Fase Integração na instituição	**Formação** **Duração:** 2 semanas Actividades de formação e integração na empresa, com as tecnologias utilizadas e com a metodologia a seguir.
Segunda Fase Projectos iniciais	**Facturação Electrónica** **Duração:** 2 meses • Análise • Desenvolvimento • Testes • Implementação no Cliente
	Knowledge Base **Duração:** 2 meses • Análise • Desenvolvimento • Testes
Terceira Fase Projecto final	**SigCRM** **Duração:** 5 meses • Análise • Desenho da Solução • Implementação • Testes

Tabela 1 Planeamento geral.

Capítulo 3

Trabalho realizado

Neste capítulo serão descritas as tecnologias utilizadas no decorrer do projecto e será feita a descrição do trabalho realizado durante o estágio.

3.1 Tecnologias utilizadas

As tecnologias envolvidas no projecto SigCRM foram a *framework* .NET 2.0 da *Microsoft*. As linguagens utilizadas foram *C#* e *JavaScript*. Para o servidor *web* foi utilizado o *Internet Information Services* (IIS) da *Microsoft*. Como plataforma para o desenvolvimento das páginas a equipa recorreu à tecnologia *ASP.NET*. O sistema de gestão de base de dados utilizado é o *SQL Server* versão 2005 da *Microsoft*.

No decorrer do projecto Facturação Electrónica foi utilizada a linguagem *VBA* para produzir a interface de ligação ao SigGC e a linguagem descritiva *XML* no formato dos documentos electrónicos.

O *Knowledge Base* será concretizado recorrendo à tecnologia *Sharepoint Services* da *Microsoft* que fornece a base para sites *web* vocacionados para a partilha de informação. O utilizador acederá ao sistema através de um *browser* de *Internet*, o que simplifica significativamente a utilização deste sistema. A base de dados estará alojada num servidor *SQL Server 2000* e o motor do servidor *web* será o *Internet Information Services* da *Microsoft*.

Comum a todos os projectos é a tecnologia utilizada para controlo de versões, o *Microsoft Visual Source Safe*.

3.2 Trabalho realizado

Como já foi referido o trabalho realizado durante o estágio dividiu-se maioritariamente entre três projectos, a Facturação Electrónica, o *Knowledge Base* e o SigCRM. Contudo, num período inicial de cerca de duas semanas teve lugar a fase de formação em que fui ambientado com as tecnologias utilizadas na empresa, assim como com a metodologia a seguir.

Fora do âmbito dos projectos mencionados neste capítulo houve espaço para desenvolver novas funcionalidades para o ERP da empresa, com especial foco na aplicação de gestão comercial SigGC.

3.2.1 Facturação Electrónica

Como já foi descrito na secção 2.1 deste relatório o projecto Facturação Electrónica consiste nos processos de importação e exportação de documentos comerciais entre duas entidades. A fase inicial de análise consistiu em recolher a informação necessária sobre a forma como se iria proceder à integração deste novo módulo na SigGC. Ainda nesta primeira fase, foram estudados os formatos da empresa proprietária para encontrar a compatibilidade ideal entre estes formatos e o modelo de dados da SigGC para o processo de importação e exportação. A segunda fase deste projecto foi dedicada ao desenvolvimento dos processos de importação e exportação de documentos descritos em seguida.

3.2.1.1 Processo de Importação de Documentos

Entenda-se por processo de importação o fluxo de dados do exterior, para a aplicação SigGC. Neste processo são importados documentos que representam notas de encomenda e/ou guias de remessa.

O processo de importação baseia-se em três componentes:

- O serviço de escalonamento de tarefas *CGScheduler*;
- O componente *ActiveX Exe* cgEBilling que actua como gestor do processo de importação;

- Uma *dll* que implementa as funcionalidades de importação que aloja as especificidades de cada importação.

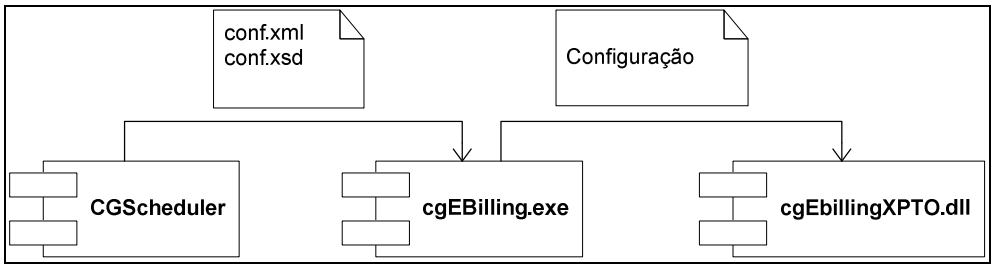

Figura 2 Facturação Electrónica - Processo de Importação

Em seguida serão descritos os três componentes do processo de importação.

- **Serviço de escalonamento *CGScheduler***

O componente *CGScheduler* é um serviço *Windows* desenvolvido pela equipa do departamento e que permite, entre outras funcionalidades, que sejam invocados objectos *COM (Component Object Model*), mais propriamente *assemblies* dentro de um objecto *COM*, com passagem de argumentos.

COM é uma plataforma da *Microsoft* para componentes de *software* usada para permitir a comunicação entre processos e a criação dinâmica de objectos em qualquer linguagem de programação. Permite a reutilização de objectos sem o conhecimento da sua implementação interna pois força o programador a fornecer uma *interface* bem definida, que está separada da implementação.

O serviço de escalonamento foi utilizado neste projecto para controlar a periodicidade das importações, assim como para assegurar a sua configuração de acordo com as necessidades de cada cliente que utilizar este módulo.

Para que este serviço possa invocar um objecto *COM*, o último necessita implementar a interface *ISchedulerAddin*. Os métodos desta interface permitem a criação de uma nova instância do objecto *COM* e a respectiva passagem dos parâmetros iniciais (*Init*), o cálculo do momento da próxima invocação (*GetNextRunDate*) e o método que executa a acção para a qual o objecto *COM* é programado (*Run*).

- **Gestor de importação cgEBilling**

Como foi referido atrás, este componente tem a responsabilidade de gerir e configurar o processo de importação. As funções deste componente incluem a criação do histórico de actividade, a configuração do processo de importação através dos ficheiros de configuração e a execução do processo de importação.

A configuração do processo de importação é feita através de um ficheiro *XML* que contém a informação sobre qual a directoria onde se encontram os documentos a importar, os dados de ligação ao servidor da aplicação SigGC, assim como a periodicidade da importação.

Devido à falta de especificação que defina univocamente o formato dos documentos electrónicos, este componente deve ser desenhado com o objectivo de ser flexível em relação às especificidades de cada formato proprietário. Posto este problema resolveu-se recorrer a padrões de desenho para desenhar uma solução reutilizável capaz de minimizar o esforço de implementação por cada formato dos documentos electrónicos que surja. Este assunto é abordado à frente na secção de opções de implementação.

- **Implementações de Importação**

Um processo de importação consiste na integração de documentos, num formato bem definido, na aplicação SigGC. O padrão de desenho *Simple Factory* foi implementado para simplificar a diversidade de formatos dos documentos electrónicos. Este componente é composto pela implementação da importação dos diversos formatos existentes. Cada formato será implementado por uma biblioteca de ligação dinâmica (*dll*).

Para pôr em prática o padrão mencionado, foi elaborada a *interface* que todas as implementações de importação devem respeitar. Esta *interface* consiste num método onde deverá ser implementada a importação e duas propriedades que permitem a passagem de configuração para a importação em formato *XML* e a sua respectiva validação através de um ficheiro *XSD*.

3.2.1.2 Processo de Exportação

O processo de exportação funciona, naturalmente, no sentido inverso do processo de importação, consistindo na criação de documentos electrónicos através de documentos existentes na aplicação SigGC. Tipicamente a exportação terá como alvo documentos que serão facturas sobre as encomendas e/ou guias de remessa importadas.

Como já foi referido na secção de objectivos deste documento, o processo de exportação foi implementado através do sistema de mapas especiais (*MS Excel*) do SigGC. Este sistema permite a integração de folhas de cálculo do *MS Excel* na aplicação SigGC.

O processo de exportação é composto por duas fases: a configuração dos documentos a exportar e a exportação dos documentos escolhidos;

A primeira fase da exportação consiste na configuração do processo.

Nomeadamente:

- O formato dos ficheiros de saída;
- O tipo de documento a exportar;
- A directoria de destino para os ficheiros
- O caminho para o ficheiro que valida os documentos exportados (ficheiro *XSD*);
- O filtro a aplicar sobre o conjunto de documentos da base de dados. Os elementos possíveis para elaborar o filtro são:
    - O código de firma;
    - O intervalo de códigos de secção;
    - O intervalo de tipos de documento;
    - O intervalo de números de documento;
    - Os intervalos de data e de vencimento do documento;
    - O intervalo de tipos de entidade;
    - O intervalo de códigos de entidade;
    - A escolha de visualizar apenas documentos que ainda tenham saldo por regularizar.

- A configuração dos dados da entidade emissora. Estes dados são preenchidos na primeira vez que o mapa é utilizado, sendo que podem ser alterados sempre que houver necessidade.

Após efectuada a configuração necessária são obtidos os documentos, finalizando a primeira fase do processo.

Na segunda fase deste processo, o utilizador tem a oportunidade de validar o conjunto de documentos obtidos a partir dos filtros definidos e, caso deseje, pode retirar documentos da lista. Após a consulta e eventual manipulação da lista de documentos a exportar, o utilizador dá inicio à exportação. Os passos do processo são registados numa folha de histórico que é apresentada ao utilizador no final do processo. Nesta folha constam os erros e as informações sobre a criação dos documentos.

3.2.1.3 Opções de Implementação

Como foi referido anteriormente na secção do processo de importação, não existe um padrão definido para o formato dos documentos electrónicos, assim sendo surgiu a necessidade de minimizar o impacto do aparecimento de um novo formato. A solução encontrada passou por recorrer à utilização de padrões de desenho.

A análise efectuada antes de pôr em prática a solução permitiu à equipa perceber que esta opção poderia ser utilizada para processos de importação de informação distinta para a aplicação SigGC.

Segue-se uma descrição dos padrões de desenho existentes, divididos pelas suas categorias com exemplos de cada uma. Posteriormente é descrito o padrão que foi utilizado na implementação do processo de importação de documentos electrónicos.

- **Padrões de Desenho**

Os padrões de desenho são a resposta à necessidade de ter soluções simples, elegantes e reutilizáveis para problemas comuns. Em seguida encontram-se algumas frases que explicam sucintamente a definição de padrões de desenho.

- "Design patterns are recurring solutions to design problems you see over and over." (*The Smalltalk Company*)

- "Design patterns constitute a set of rules describing how to accomplish certain tasks in the realm of software development." (Pree 1994)
- "Design patterns focus more on reuse of recurring architectural design themes, while frameworks focus on detailed design and implementation." (Coplien and Schmidt 1995)
- "A pattern addresses a recurring design problem that arises in specific design situations and presents a solution to it." (Buschmann et al. 1996)
- "Patterns identify and specify abstractions that are above the level of single classes and instances, or of components." (Gamma et al. 1993)

Os padrões de desenho ganharam reconhecimento formal no início dos anos noventa através de Erich Gamma (1992). Este processo culminou com a publicação do livro "Design Patterns – Elements of Reusable Software", pela mão de Gamma, Helm, Johnson e Vlissides (1995). Esta publicação cobre vinte e três padrões recorrentes, apresentando as respectivas soluções e casos de aplicação.

Na publicação supracitada os padrões de desenho foram divididos em três categorias, descritas em seguida.

- ***Padrões de Criação***: focam-se na forma como se criam as instâncias dos objectos, são uma alternativa à instanciação directa destes. Permitem a uma aplicação ser flexível no processo de decisão sobre que objecto é necessário numa determinada situação. Estes padrões são úteis na medida em que uma aplicação não deve ser dependente da forma como os objectos são criados. São exemplo desta categoria o *Factory Pattern*, utilizado para escolher e retornar uma instância de um grupo de objectos semelhantes, com base nos dados fornecidos. O padrão *Abstract Factory* utilizado para retornar um grupo de um conjunto de grupos de objectos. Por vezes o retorno pode ser um *Factory* para um grupo. O padrão *Builder* que tem como objectivo associar um conjunto de objectos de determinada forma, calculada através dos dados fornecidos, criando um novo objecto. É prática corrente utilizar uma *Factory* para determinar a forma como os objectos são associados. Em seguida temos o padrão *Prototype* que ao invés de criar uma nova instância de um objecto, retorna uma cópia ou um clone de um objecto existente, evitando assim o *overhead* da criação de uma nova instância. Finalmente temos o padrão *Singleton* que permite

assegurar que existe uma e só uma instância de um determinado objecto e que esta é globalmente acessível.

- ***Padrões Estruturais***: descrevem como se podem agrupar objectos para formar estruturas. Os padrões descritos sumariamente em seguida são exemplos desta categoria. O padrão *Adapter* pode ser utilizado com o objectivo de compatibilizar duas *interfaces*, assim, um objecto que implemente uma *interface* pode comunicar com outro objecto que implemente uma *interface* diferente. Este processo obtém-se tipicamente por um de dois processos, herança e composição de objectos. O padrão *Composite*, que é utilizado para a criação de conjuntos de objectos, em que, alguns deles podem ser também compostos. Ainda nesta categoria temos o padrão *Proxy* em que um objecto simples é utilizado no lugar do original, mais complexo, que pode ser instanciado posteriormente. *Flyweight* é um padrão utilizado para a partilha de objectos, em que cada instância não contém o seu próprio estado – ao invés, este é armazenado externamente. Este padrão permite uma partilha eficiente de objectos quando existem muitas instâncias mas poucos tipos diferentes. O padrão *Façade* é utilizado de forma a esconder a complexidade de um subsistema numa *interface* que mantém toda a funcionalidade, embora à custa de alguma flexibilidade. O padrão *Bridge* que tem o objectivo de separar a *interface* de um objecto da sua implementação, de forma a permitir que estes possam variar separadamente. Finalmente temos o padrão *Decorator* utilizado quando é necessário que as responsabilidades de um objecto sejam atribuídas dinamicamente.

- ***Padrões Comportamentais***: têm o propósito de definir a comunicação entre objectos e o fluxo de informação, entre estes, numa aplicação. Nesta categoria estão incluídos os seguintes padrões. O padrão *Chain of Responsability* que permite um maior grau de independência entre objectos através da passagem de pedidos, que um objecto não consegue atender, para o seu sucessor na cadeia de objectos. *Command* é o padrão que tem como objectivo que cada acção dentro de uma aplicação seja executada por apenas um objecto que implementa uma determinada *interface*, permitindo desta forma que o cliente desconheça por completo a acção que vai executar e permite que a implementação da acção seja modificada sem a necessidade de reestruturação do código cliente. O padrão *Iterator* estabelece a forma como a aplicação percorre uma lista de dados dentro de um objecto, através de uma *interface* sem a necessidade de conhecer os detalhes da representação interna dessa lista. O

padrão *Mediator* define a forma como a comunicação entre objectos pode ser simplificada através da utilização de um objecto intermediário com o objectivo de dispensar que todos os objectos se conheçam entre si. *Memento* é o padrão que permite concretizar a funcionalidade de um objecto guardar o seu estado num dado momento no tempo para o restaurar mais tarde. O padrão *Observer* define a forma como um conjunto de objectos reflectem a mudança de informação da qual todos dependem. O padrão *State* que permite a um objecto modificar o seu comportamento de acordo com o seu estado interno. O padrão *Strategy* permite que uma acção seja executada através de um objecto por um algoritmo do conjunto dos algoritmos capazes de executar essa acção. A escolha do algoritmo pode ser feita com intervenção do utilizador, ou por um gestor de contexto. O padrão *Template Method* é utilizado através do mecanismo de herança, por exemplo, na implementação de um algoritmo comum em que alguns passos podem ser executados de forma distinta; assim, deixa-se essa implementação para subclasses distintas. Finalmente, o padrão *Visitor* tem a capacidade de adicionar novas funcionalidades a um objecto sem a necessidade de o alterar. Através do polimorfismo, este padrão garante o acesso a diferentes objectos sem a necessidade de alteração de código.

- **Padrão Simple Factory**

Este padrão de desenho é uma aproximação simplificada do padrão *Factory Method* sumariamente descrito atrás. Um *Simple Factory* devolve uma instância de um conjunto de objectos possíveis, com base na informação fornecida. É comum que todos os objectos do conjunto implementem a mesma interface ou herdem do mesmo objecto, mas que, contudo executem as suas tarefas de modo distinto e/ou optimizado para diferentes tipos de dados.

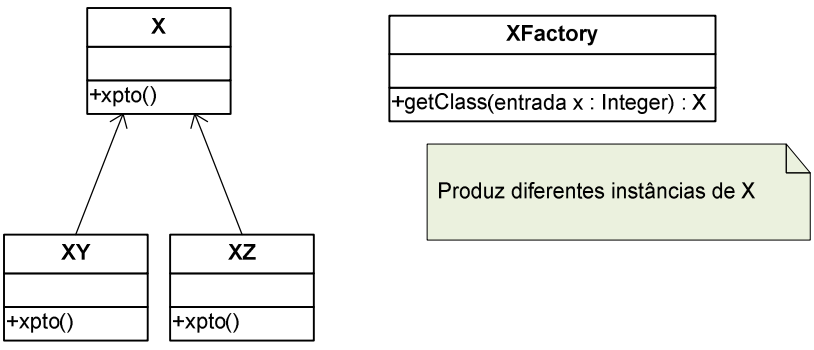

Figura 3 Padrão *Simple Factory*

Na figura 3, *X* é uma classe base, e as classes *XY* e *XZ* são derivadas da primeira. A classe *XFactory* tem a responsabilidade de decidir qual a subclasse do objecto a retornar, com base nos argumentos fornecidos. Na classe *XFactory* é definido o método *getClass* que recebe um parâmetro de entrada e retorna uma instância de *X*. O resultado deste método é ser transparente para o programador, uma vez que ambas as subclasses têm os mesmos métodos, apenas diferindo na sua implementação. A forma como o método decide o seu retorno é da responsabilidade da *factory*, podendo ser obtido através do cálculo de uma função extremamente complexa, ou simples.

3.2.2 *Knowledge Base*

Este projecto tem como objectivo criar um repositório de conhecimento que permita acelerar o processo existente que decorre entre a entrada de uma nova ocorrência pela equipa de assistência técnica do departamento e a sua solução através da equipa de desenvolvimento.

O sistema baseia-se numa arquitectura cliente/servidor na qual o servidor recorre ao *Internet Information Services* para disponibilizar as páginas *web*, cuja estrutura tem origem na ferramenta *Sharepoint Services* da *Microsoft*, e ao *SQL Server 2000* para manter a base de dados com o conteúdo do repositório de conhecimento.

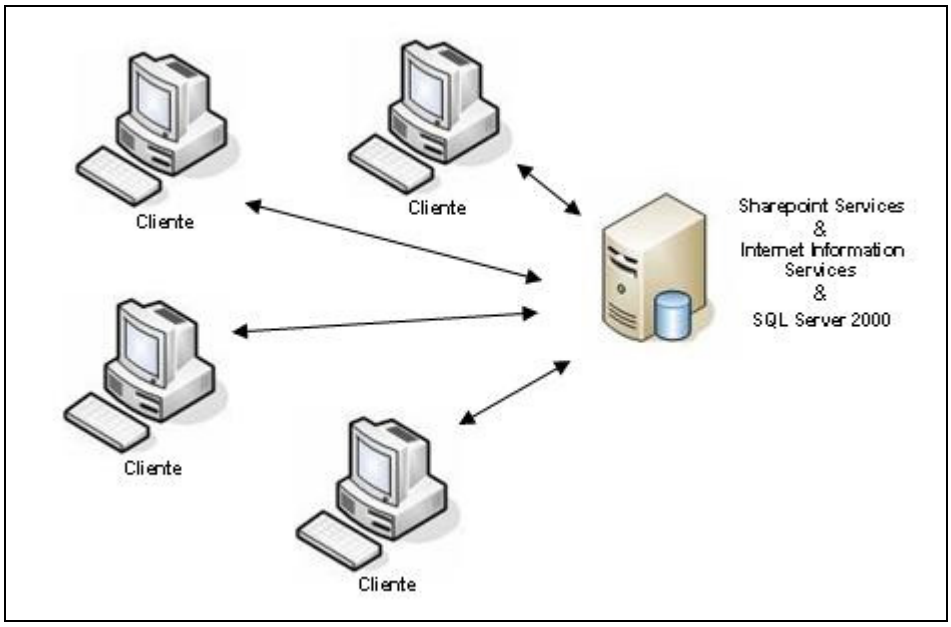

Figura 4 Vista geral do sistema do Knowledge Base

Neste sistema o fluxo de informação é iniciado pelo utilizador no momento em que acede à página de entrada do repositório. Todas as manipulações e operações de pesquisa são enviadas para o servidor através do *browser* e é através deste que o utilizador recebe a resposta às suas acções. As consultas processam-se exactamente da mesma forma: o utilizador inicia o ciclo e o servidor responde através do *browser*.

Em seguida serão descritos os conceitos do sistema, que no seu total são a base das funcionalidades do *Knowledge Base*.

3.2.2.1 Sistema de Acessos

Existe a necessidade de restringir as funcionalidades existentes por grupos de utilizadores, de modo a que possam ser distribuídas responsabilidades diferentes consoante o utilizador.

Existem quatro grupos de utilizador no repositório. São eles:

- **Leitor:** tem acesso só de leitura ao *web site*;
- **Contribuinte:** pode adicionar conteúdo a bibliotecas de documentos e listas existentes;
- **Designer da Web:** pode criar listas e bibliotecas de documentos, assim como personalizar páginas no *web site*;
- **Administrador:** tem controlo total sobre o *web site*.

3.2.2.2 Conceito de Ocorrências

O conceito de ocorrência foi criado para representar os acontecimentos que surgem no decorrer da utilização das aplicações da empresa. Uma ocorrência, tal como é definida neste contexto, é algo que requer análise.

A análise é necessária para que uma ocorrência possa ser categorizada e processada para se atingir uma resolução.

A concretização de uma ocorrência no ambiente do repositório é alcançada através da criação de um item que contém atributos. Entre outras informações os atributos representam o título da ocorrência (*Título*), o seu autor e a data em que foi criada (*Reportado por*, *Dia em que foi reportado*), o estado de uma ocorrência (*Estado*), os passos para a sua resolução (*Resolução*) e a aplicação com que está relacionada (*Aplicação*).

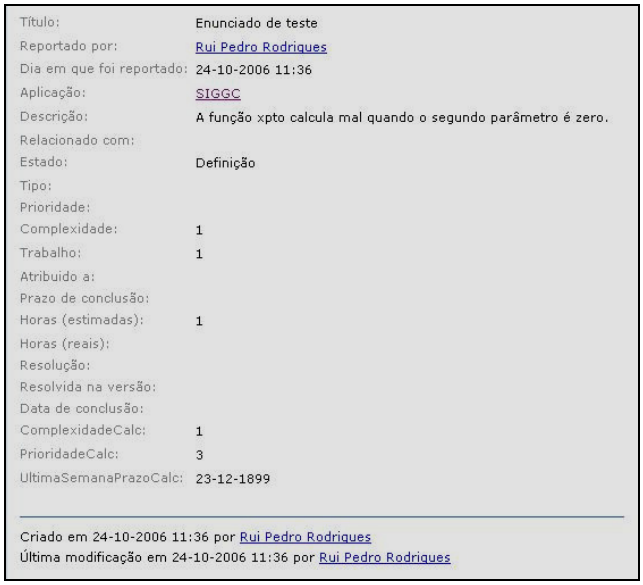

Figura 5 Knowledge Base - Atributos de uma ocorrência

As ocorrências estão agrupadas num conjunto; esse conjunto é concretizado no repositório através de uma lista à qual podem ser aplicadas vistas e/ou filtros de forma a exibir apenas as ocorrências desejadas e sobre a qual se podem criar peças *web* para inserir nas páginas do site. Os conceitos que aparecem neste parágrafo serão todos descritos nesta secção.

3.2.2.3 Conceito de Lista

A lista é um dos componentes disponibilizados pelo *Sharepoint* que é usado para modelar o contexto do repositório. Uma lista é simplesmente um conjunto de itens, por exemplo a lista *Ocorrências* é um conjunto de itens do tipo ocorrência (ver 3.2.2.2).

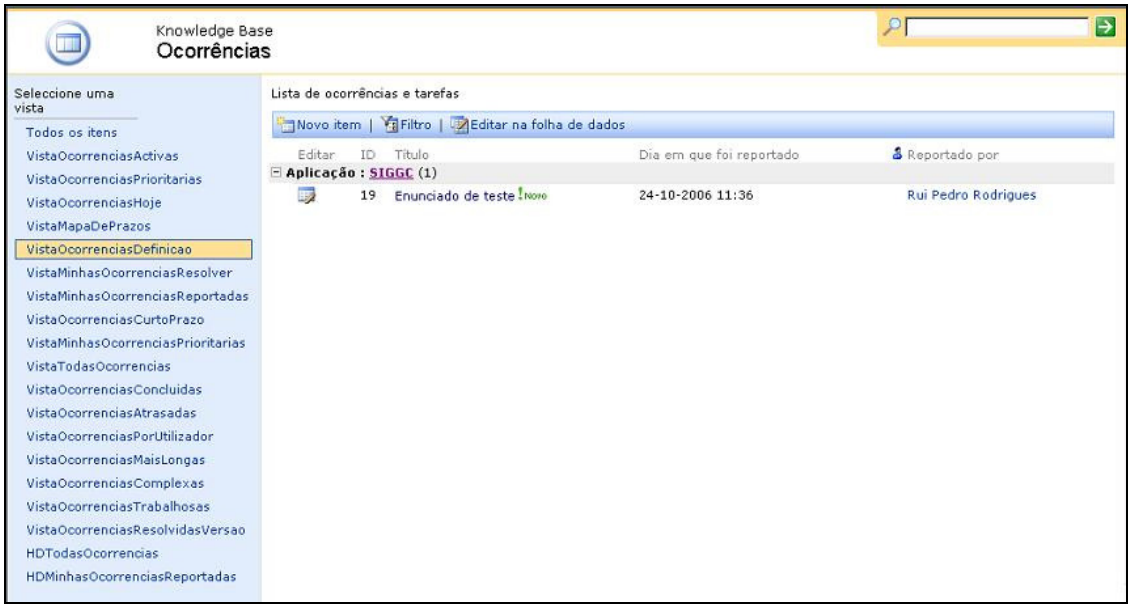

Figura 6 Knowledge Base – Página de Lista de Ocorrências

Dada a sua natureza, uma lista tem de fornecer meios para tornar a sua pesquisa eficiente. Para esse efeito podem ser aplicadas vistas sobre uma lista, assim como é possível filtrar uma lista por um ou mais campos dos itens. As vistas e filtros são úteis para a procura do tipo "agulha no palheiro" mas a sua maior vantagem revela-se na possibilidade de exibir apenas partes específicas de uma lista, como por exemplo "Todas as ocorrências relacionadas com a aplicação SigGC".

3.2.2.4 Conceito de Biblioteca de Documentos

Tal como o conceito definido anteriormente, a biblioteca de documentos é também um componente fornecido pelo *Sharepoint*, utilizado na modelação do repositório. Como o próprio nome indica este componente pretende ser uma biblioteca que pode ser pesquisada através das mesmas ferramentas descritas para as listas.

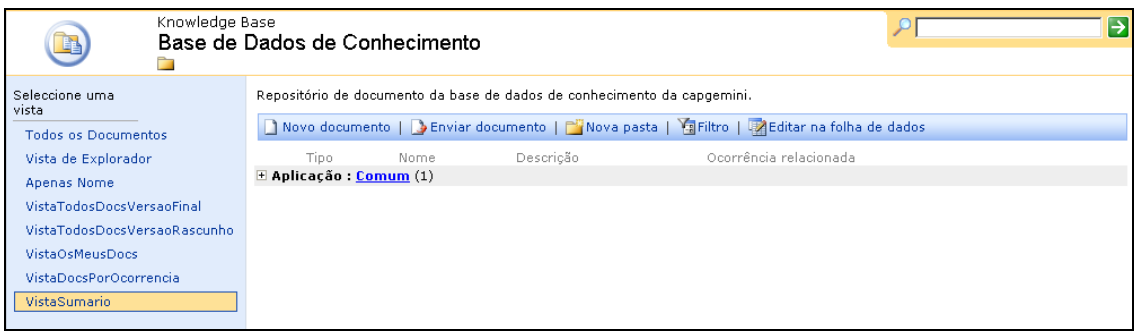

Figura 7 Knowledge Base - Página da Base de Dados de Conhecimento

Analogamente à lista, a biblioteca de documentos também contém itens, mas neste caso são de qualquer tipo de documento que o utilizador importe. O item documento possui também um conjunto de atributos que permitem categorizá-lo e tornar as pesquisas mais eficientes.

3.2.2.5 Conceito de Vista

O conceito de vista não é tão transparente como o de lista (ver 3.2.2.3), todavia explicando o seu propósito, o conceito torna-se claro.

Uma vista é aplicada sobre um componente para que este exiba apenas os itens que se enquadrem no critério definido para essa vista. Com esta definição do objectivo de uma vista surge o conceito de critério de uma vista. O critério é o conjunto de valores das propriedades de uma vista.

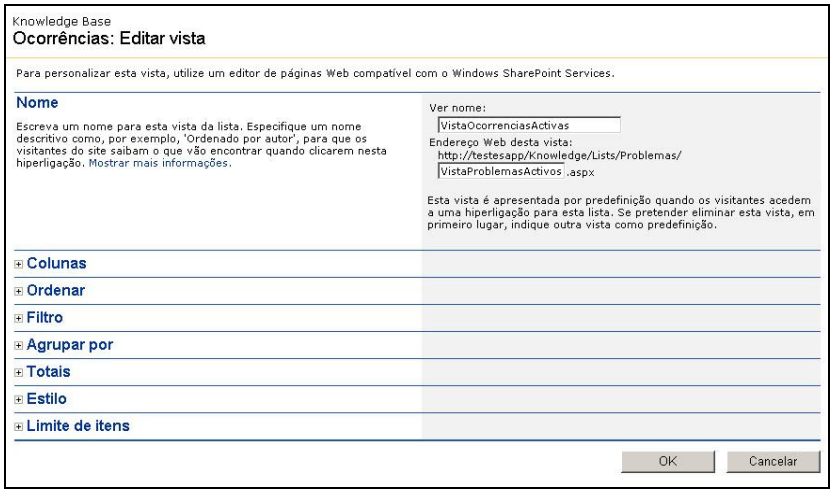

Figura 8 Knowledge Base - Página de edição de uma vista

Como é possível observar na figura 8 uma vista é composta por diferentes atributos, entre os quais o nome que permite a sua identificação, o conjunto de colunas que deverão ser exibidas, cláusulas de ordenação e agrupamento entre outros.

Existe um conjunto de vistas definido por defeito que pode ser consultado na secção 8.3 deste documento.

3.2.2.6 Conceito de Peça *Web*

Uma peça *Web* é semelhante a uma moldura que reveste os elementos do repositório, permitindo ao utilizador "espreitar" para o interior desses elementos.

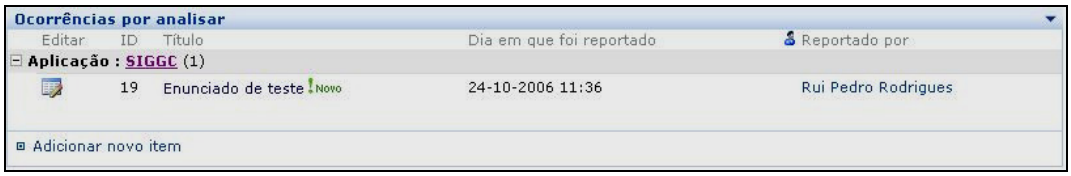

Figura 9 Knowledge Base - Peça Web «Ocorrências por analisar»

A peça *Web* tem sempre um componente de base e uma vista para esse componente. No exemplo da figura 9 a peça *Web* tem o título *Ocorrências por analisar*, tem como base o componente *Lista de Ocorrências* e a vista seleccionada é *VistaOcorrênciasDefinição*. Como foi dito, uma peça tem um título e tem também uma vista associada, estes são dois dos seus atributos incluídos no conjunto dividido por categorias: aspecto, esquema e avançadas.

3.2.2.7 Conceito de Página de Componente

Como já vimos nas secções anteriores, o *Sharepoint* fornece componentes para a modelação de contextos. Cada componente tem uma página *web* a partir da qual pode ser configurado ou pesquisado e essa página designa-se, neste documento, de página de componente[1].

Através da página de componente conseguimos aceder ao menu de configuração do componente ou percorrer de forma rápida todas as vistas disponíveis para esse componente.

[1] A figura 6 é o exemplo da página de componente da Lista de Ocorrências.

Existem diversas formas para aceder à página de um componente – a título de exemplo, uma dessas formas é seguir a ligação existente entre o título de uma peça *web* criada com base num componente e a página do próprio.

3.2.2.8 Conceito de Página de Categoria

No site do repositório de conhecimento existem páginas que neste documento são designadas por páginas de categoria. Uma página de categoria tem o propósito de fornecer um conjunto de peças *Web* (ver secção 3.2.2.1.6) com vistas úteis sobre os componentes existentes, para um conjunto de utilizadores. Por exemplo, inicialmente existem quatro páginas de categoria:

- *Desenvolvimento*: vocacionada para a equipa de desenvolvimento;
- *Gestão*: com vistas focadas na gestão de ocorrências e documentos;
- *HelpDesk*: para a equipa do *help desk* e outros;
- *Pessoal*: é uma página com vistas sobre as ocorrências e os documentos do utilizador actual.

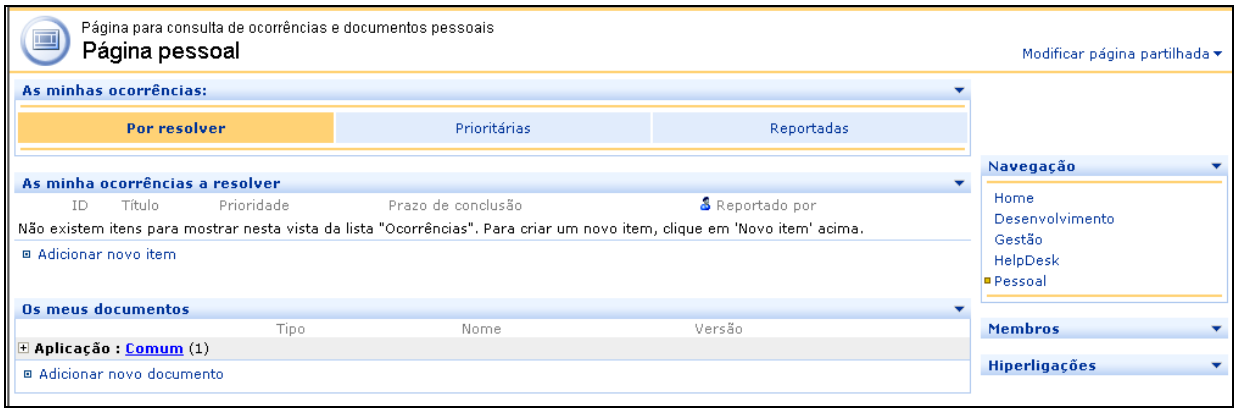

Figura 10 Knowledge Base - Página de Categoria (*Pessoal*)

3.2.2.9 Conceito de Página de Entrada

A página de entrada é aquela a que o utilizador acede imediatamente após ter sido identificado com sucesso no repositório.

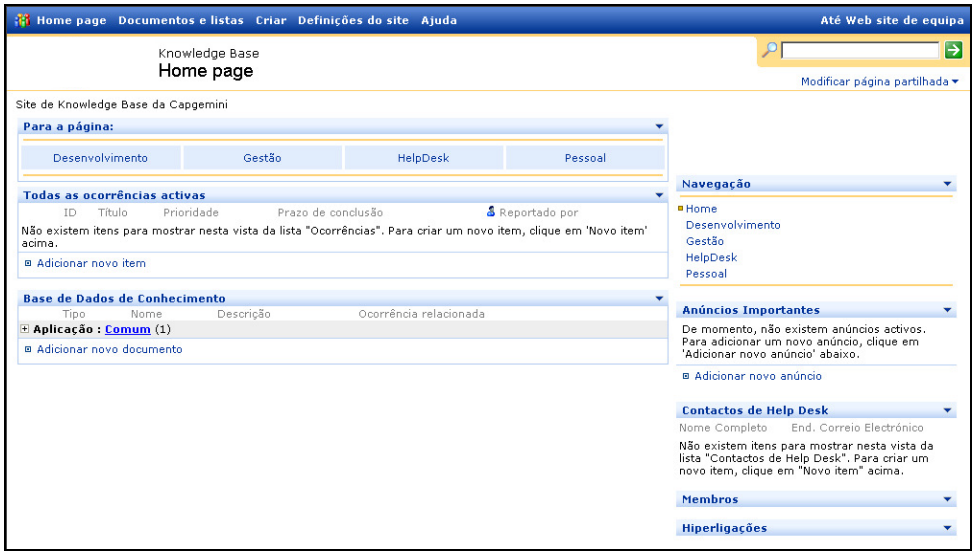

Figura 11 Knowledge Base - Página de entrada do repositório de conhecimento

3.2.3 SigCRM (Sistema Integrado de Gestão - *Customer Relationship Management*)

Como já foi descrito anteriormente, uma aplicação de CRM permite elevar o nível de proximidade entre clientes e fornecedores/prestadores de serviços, por exemplo, através do acompanhamento personalizado e do registo de necessidades. Existem também vantagens na vertente administrativa de uma aplicação de CRM, como por exemplo, a possibilidade de medir o grau de eficiência da equipa de comerciais da empresa, o registo dos contactos efectuados com os clientes, assim como os relatórios de visita aos mesmos. Nesta secção será descrito o trabalho realizado neste projecto.

3.2.3.1 Técnicas e Ferramentas utilizadas na aplicação

Esta secção é dedicada à descrição de algumas das técnicas utilizadas no desenvolvimento da aplicação, assim como à descrição de ferramentas, desenvolvidas pela equipa e que foram utilizadas no desenvolvimento do projecto.

- **Cookies**

Um *cookie* é um par do tipo *nome/valor* armazenado na máquina cliente. Esta técnica é utilizada por defeito pelo ASP.NET para manter o estado de uma sessão. No entanto podem ser utilizados *cookies* para armazenar, de forma temporária ou persistente, informações sobre o utilizador, como por exemplo os seus dados, ou o estado de uma operação efectuada. Os *cookies* estão acessíveis através de uma instância da colecção *HttpCookieCollection*. Os objectos *HttpRequest* e *HttpResponse* ambos têm uma instância da colecção referida, e significam os *cookies* enviados do cliente para o servidor, no primeiro caso e os *cookies* que serão enviados do servidor para o *browser* no segundo.

Existem os *cookies* de sessão, que são temporários e uma vez terminada a sessão o seu conteúdo perde-se. Em alternativa existem os *cookies* persistentes, que são fisicamente armazenados no disco do cliente, sendo o seu conteúdo preservado mesmo quando a sessão termina. Em ambos os casos uma vez enviado para o *browser*, o *cookie* é sempre enviado para o servidor em cada pedido *Http*.

- **URL encoding**

URL encoding é uma técnica alternativa ao uso de *cookies* que permite armazenar informação no *url* da página a exibir. A informação é guardada numa *query string* que é concatenada no final do *url*. Esta técnica permite o transporte de informação entre páginas. No SigCRM é aplicada, por exemplo, quando se efectua uma consulta à lista de oportunidades e em seguida se edita um registo, o identificador da oportunidade em edição (*http://localhost/CRM/Oportunidade.aspx?Id=18*) é passado no *url*, entre outras informações.

- **Acessos/Permissões e Sessões**

Nesta secção será descrito o método utilizado para manter o estado da sessão na aplicação e será abordado o sistema de acessos da mesma.

Para manter o identificador de uma sessão do utilizador (*session tracking*) são utilizados *cookies*, cuja informação é cifrada para obter maior grau de segurança. Ainda a nível da segurança a opção por utilizar *cookies* para manter o identificador de sessão invés do método *cookieless* deve-se ao facto de neste último método o identificador ser passado no *URL* da página. Ainda que codificado este método fragiliza a segurança, na medida em que o identificador de sessão "viaja" na rede, permitindo que ataques de escuta de rede obtenham este valor.

Outro dos motivos pelo qual foi utilizado este método está relacionado com o facto de o conceito de sessão ter sido implementado de forma independente da janela do *browser*, isto é, a sessão mantém-se mesmo que o utilizador feche a janela do *browser* que contém a aplicação. A forma de implementação baseia-se numa tabela no servidor que mantém um identificador de sessão (independente da sessão do *browser*) que, caso o utilizador deseje, consegue recuperar a sessão na qual o utilizador estava a trabalhar anteriormente. Caso se verifique que a sessão expirou o utilizador é redireccionado para a página de entrada.

No que diz respeito à gestão de acesso a certas páginas, a aplicação mantém um conjunto de papéis de utilizador (*roles*), cada um com um conjunto de permissões. Quando um utilizador tenta executar uma acção, a aplicação verifica se este tem acessos suficientes de acordo com o seu papel na aplicação e permite ou não a execução dessa acção.

- ***Framework***

Foi utilizada uma *framework*, desenvolvida pela equipa do departamento, durante a fase de implementação deste projecto.

A *framework* disponibiliza diversas funcionalidades, das quais constam:

1. Gestão da *cache*: contém as preferências de cada utilizador;
2. Gestão da configuração da aplicação: definições da aplicação que podem ser modificadas pelo utilizador;
3. Gestor de contexto: tem a capacidade de gerir o contexto de um utilizador, de uma entidade e de uma base de dados dentro de uma aplicação;
4. Gestor de recursos: permite internacionalizar as aplicações através da utilização de recursos;
5. Segurança: fornece classes que disponibilizam funcionalidades de segurança, como por exemplo: cifrar/decifrar informação e efectuar cálculo de *hash* sobre blocos de informação;
6. Utilidades: classes que auxiliam na manipulação de informação em formato *XML*, na manutenção do registo de *log* da aplicação, na manutenção da auditoria dos registos de entidade;
7. Motor de Mapas: capaz de gerar *reports* personalizados com informação de uma aplicação;
8. Avaliador de Código: é uma ferramenta capaz de compilar e executar código em *runtime*, o que permite a capacidade de *scripting* dentro das aplicações;
9. Motor de Regras: tem a capacidade de implementar um conjunto de regras, validá-las e executar acções em função desse resultado;
10. Modelador de Base de Dados (*DAL*): tem a capacidade de modelar em objectos as tabelas, *views* e *constraints* de uma determinada base de dados.

3.2.3.2 Funcionalidades implementadas

Nesta secção serão descritas as funcionalidades que desenvolvi para o projecto SigCRM. O meu trabalho foi principalmente desenhar as classes base para as páginas da aplicação, para os *server controls* e implementar a funcionalidade de manutenção das tabelas de dados da aplicação, como por exemplo a manutenção da tabela de estados possíveis de um contrato. Ainda nesta secção é descrita a tecnologia *Ajax*, utilizada em alguns dos *server controls* da aplicação e a forma como esta foi introduzida na aplicação.

I. Classes base

Uma das tarefas que realizei foi o desenho das classes base que suportam as páginas e alguns dos *controls* existentes na aplicação; nesta secção encontram-se descritas essas classes.

- **Estrutura das páginas**

A classe base de todas as páginas da aplicação é a "*PageBase*" (ver figura 12). Esta classe herda de "*System.Web.UI.Page*" e tem a responsabilidade de gerir o intervalo de validade de uma sessão, gerir o acesso e a verificação de permissões de um utilizador e ainda a gestão da barra de estado existente no cabeçalho da página.

Descendo um nível na especificação, temos as páginas de lista de conceito e as páginas de conceito. Uma página de lista de conceito herda de "*ListPageBase*" e o seu propósito é conceder a possibilidade de pesquisa sobre um conjunto de elementos de um conceito da aplicação como por exemplo, a lista de oportunidades de venda. Ainda neste tipo de página constam o conjunto de filtros passíveis de serem aplicados no critério de pesquisa, assim como o conjunto dos resultados da pesquisa efectuada. Este tipo de página tem a responsabilidade de gerir a informação sobre o seu título, a origem dos seus dados e a capacidade de validar o acesso aos elementos do resultado da pesquisa.

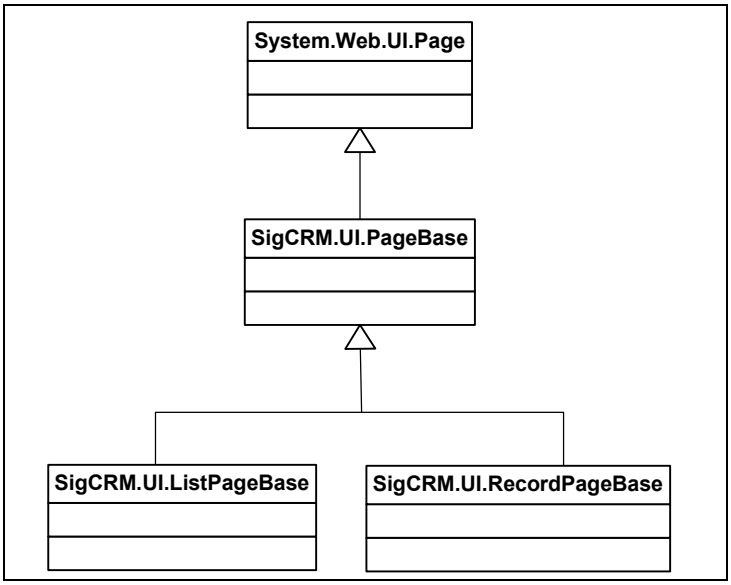

Figura 12 Hierarquia das classes base das páginas do SigCRM.

Uma página de conceito herda de "*RecordPageBase*" e o seu objectivo é exibir um conceito existente na aplicação, seguindo o exemplo dado para as páginas de lista; numa página de conceito teríamos os dados de uma oportunidade de venda. As funcionalidades presente neste tipo de página são a gestão do seu título, a capacidade de gerir o modo de manipulação do conceito, por exemplo, inserção, edição e eliminação, a respectiva validação das permissões necessárias por parte do utilizador para cada acção e ainda a origem e destino dos seus dados.

A classe "*PageHeaderBase*" é também uma classe base que permite às suas extensões a criação de um cabeçalho na página que incorpore *controls* para centralizar as acções relativas ao conteúdo da página. No exemplo da figura 13, o cabeçalho contém o título da página actual (Alterar Entidade) à esquerda e os botões que permitem Gravar e Eliminar o registo actual ou voltar à página anterior, à direita.

Figura 13 SigCRM - Exemplo de cabeçalho de página, extensão da classe "*PageHeaderBase*".

É da responsabilidade da classe que estende "*PageHeaderBase*" implementar o comportamento dos *controls* que escolhe exibir no cabeçalho.

- **LOV controls**

Foi necessário desenhar uma classe base para *controls* da aplicação que permitem efectuar pesquisa sobre a base de dados. Estes têm como base a classe *LovControlBase* e esta classe por sua vez tem como base a classe *System.Web.UI.UserControl*.

A principal funcionalidade deste tipo de *control* é permitir a pesquisa de registos na base de dados. Após a selecção de um registo o *control* fica associado ao identificador desse registo, facilitando as acções desejadas pela aplicação.

Figura 14 SigCRM - Exemplo de um LOV *control*.

É possível implementar as acções de inserção de um novo registo, associação de dois registos de tabelas diferentes (relacionados por chave estrangeira), e é também possível implementar a forma como o *control* volta ao seu estado original (*clean*).

As extensões de *LovControlBase* têm a responsabilidade de definir o comportamento adequado para a inserção, associação e retorno ao estado original do *control*, providenciar a fonte de dados para a pesquisa, assim como a *query* que essa pesquisa efectua e ainda o elemento *div* que contém o aspecto e conteúdo do painel de pesquisa a exibir.

II. *Ajax (Asynchronous Javascript And XML)*

Nesta secção será retratada a implementação da tecnologia *Ajax* neste projecto. *Ajax* é a terminologia atribuída a um conjunto de ferramentas existentes. O centro desta tecnologia é a classe *XMLHttpRequest*.

O propósito de recorrer a esta tecnologia centra-se nas vantagens de construir *websites* mais rápidos, dinâmicos e ao mesmo tempo se consigam poupar recursos. É também possível obter a melhoria da partilha de recursos através do aproveitamento do poder de processamento dos clientes. Ao invés de delegar essa responsabilidade exclusivamente no servidor, as máquinas dos clientes passam a executar processamento (*Javascript*) com os dados obtidos do servidor. A poupança de recursos é alcançada devido ao facto de não ser o servidor a processar a totalidade das páginas através de *web services* ou *scripts PHP* (todo o conteúdo das páginas era enviado através da rede). O uso desta

tecnologia permite a actualização selectiva de partes do conteúdo de uma página que pode conter imagens, documentos e/ou menus, e não da sua totalidade. Por exemplo, pode-se efectuar o processamento dos dados nos campos de um formulário e o seu resultado ser imediatamente exibido na página.

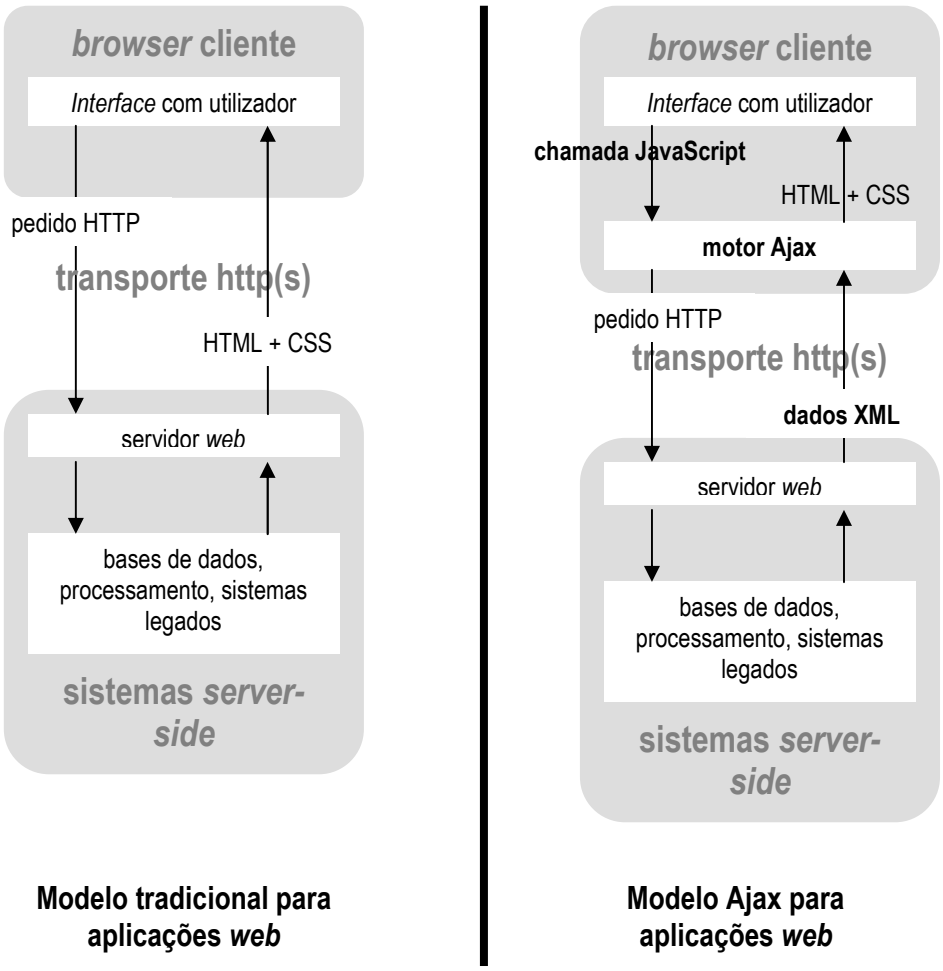

Figura 15 O modelo tradicional para aplicações *web* (à esquerda), comparado com o modelo Ajax (à direita).

O modelo tradicional das aplicações *web* (na Figura 15, à esquerda) funciona da seguinte forma: a maioria das acções do utilizador desencadeia um pedido *HTTP* para o servidor. O servidor, por sua vez, efectua o processamento necessário e em seguida retorna para o cliente uma página *HTML*. O principal defeito para este modelo reside na sua pobreza ao nível da experiência proporcionada ao utilizador isto porque, enquanto o servidor está a fazer o seu trabalho, o utilizador não pode fazer nada a não ser esperar.

A principal diferença no novo modelo (Figura 15, à direita) é a introdução da camada do motor *Ajax*, capaz de suprimir as deficiências de interacção existentes no modelo tradicional. Ao iniciar uma sessão o *browser* carrega um motor *Ajax*, ao invés de carregar uma página. Este motor é responsável pela apresentação do *interface* que o utilizador visualiza, assim como pela comunicação com o servidor. Através desta camada o utilizador consegue interagir com a aplicação num modo assíncrono, independente da comunicação com o servidor.

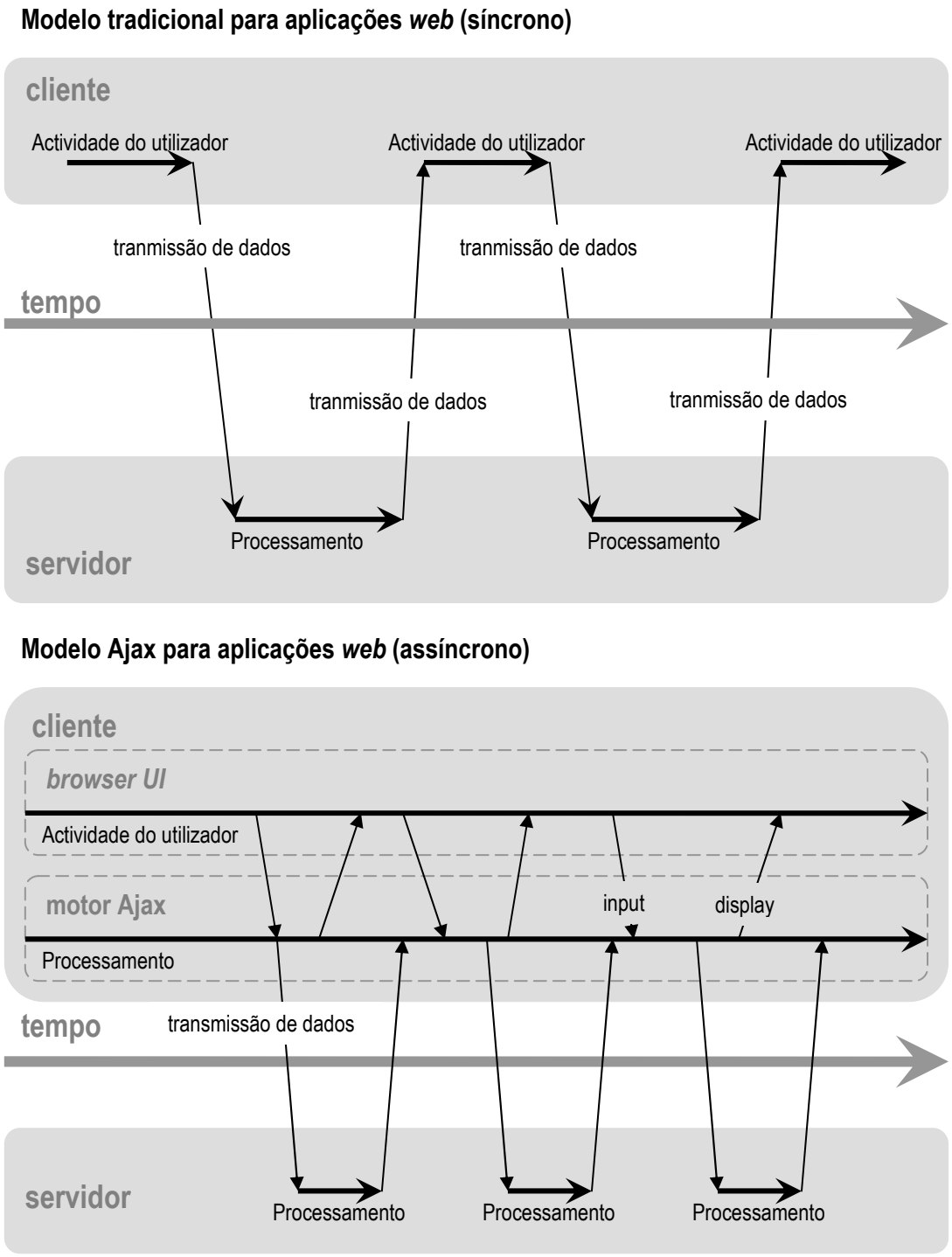

Figura 16 Padrão síncrono do modelo tradicional das aplicações *web* (em cima) comparado com o modelo assíncrono de uma aplicação Ajax (em baixo).

Cada acção do utilizador, que no modelo tradicional origina um pedido *HTTP* para o servidor (Figura 16, em cima), toma a forma de uma chamada *JavaScript* ao motor *Ajax* no modelo assíncrono. A qualquer acção do utilizador que não requeira processamento do servidor, como por exemplo validações, edição de dados em memória ou algumas formas de navegação na página, o motor consegue responder sem acesso ao servidor.

Contudo, caso o motor necessite de aceder ao servidor para elaborar a resposta ao pedido do utilizador, como por exemplo o envio de dados para o servidor, carregar código adicional para a *interface* ou obter novos dados, este efectua os pedidos de forma assíncrona, recorrendo normalmente a *XML*, sem bloquear a interacção do utilizador com a aplicação (Figura 16, em baixo).

- ***Framework Anthem***

Anthem é a *framework* de auxílio à implementação das funcionalidades *Ajax* que foi utilizada neste projecto. O seu funcionamento é relativamente simples e consiste no seguinte. Esta *framework* contém um conjunto de *server controls* que estendem os *server controls* de *ASP.NET*, mantendo o modelo de programação existente incluindo suporte total para *viewstate* e eventos *server-side*; para a sua utilização é necessário apenas a substituição dos *controls* escolhidos pelos da *framework.*

- ***Ajax* no SigCRM**

O modelo *Ajax* descrito nas secções anteriores está presente no SigCRM nos *controls web* da *framework*. Estes *controls* estendem os respectivos *server controls* da *framework Anthem*.

III. Manutenção de Tabelas

A manutenção de tabelas consiste em permitir que um utilizador, com determinados privilégios, tenha a capacidade de efectuar acções de manutenção sobre as tabelas do modelo de dados do SigCRM que, de outro modo, não são acessíveis como por exemplo, a tabela de estados de um contrato.

Esta funcionalidade está acessível através do menu de administração do SigCRM, nos sub menus "Manutenção de Tabelas I" e "Manutenção de Tabelas II". Os *links* dos elementos destes menus são gerados automaticamente. Os elementos destes sub menus podem ser adicionados no *control* do menu (*Menu.ascx*), da aplicação.

Ao aceder a um dos elementos deste menu, a aplicação gera uma página constituída por uma grelha (*GridViewUC*) com as colunas da tabela respectiva e com os *controls* que

permitem editar, gravar ou apagar um registo. À diferença das restantes grelhas da aplicação, a *GridViewUC* permite a edição do seu conteúdo nas células.

A construção das colunas da grelha é feita dinamicamente através do *xml* devolvido pela classe que representa a tabela no *DAL*. O *DAL* é uma biblioteca de classes geradas automaticamente com o objectivo de representar as tabelas e *views* do modelo de dados da aplicação.

IV. Separadores (*Tabs*)

Esta funcionalidade consiste em colocar no fundo de cada página um conjunto de separadores com áreas de informação relacionada com o conteúdo da página. Por exemplo, a página de uma oportunidade contém os separadores Produtos, Processos, Tarefas, Contactos, Relatórios de Visita, Documentos, Observações e Auditoria. Cada um destes separadores permite que seja associado um registo desse conceito ao conceito da página (por exemplo, associar um produto a uma oportunidade).

Os separadores são adicionados no fundo da página na tabela de separadores onde cada separador contém um *LinkButton* para exibir os seus *controls*. Cada separador contém, numa grelha, a actual lista de registos associados e os botões que permitem criar ou remover uma associação.

Os *controls* existentes em cada separador estendem a classe *LOVControlBase*. As acções disponíveis em cada separador são implementadas por um *LOV control*. Tipicamente essas acções consistem na possibilidade de associar um novo registo do conceito do separador ao registo principal da página e inserir um novo registo do conceito do separador para este ser associado em seguida.

Capítulo 4

Conclusão

4.1 Sumário e crítica do trabalho realizado

Este relatório é relativo ao trabalho realizado no âmbito do Projecto em Engenharia Informática inserido no Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa. O trabalho realizado decorreu na Capgemini Portugal e consistiu no desenvolvimento de três projectos – a implementação do módulo de facturação electrónica na aplicação de gestão comercial do ERP da empresa, a criação do repositório de conhecimento do departamento de aplicações *standard* da empresa e, finalmente, a participação no projecto SigCRM.

O projecto da facturação electrónica consistiu em implementar o mecanismo de importação e exportação de documentos electrónicos a ser integrado na aplicação SigGC.

As principais dificuldades que encontrei neste projecto foram a falta de experiência a nível de negócio. A título de exemplo refiro a minha falta de noções sobre a articulação e fluxo de documentos numa actividade comercial, por exemplo, o circuito estabelecido entre um orçamento, passando por uma guia de remessa e terminando na conhecida factura era-me totalmente estranho. Depois, a parte associada ao respectivo seguimento do processo com o pagamento da factura, a contabilização do pagamento e a sua divisão por centros de custo, a parte da tesouraria e tudo mais que, na minha opinião são noções que pecam por escassas nos cursos mais gerais do ramo da engenharia informática.

A um nível mais técnico as minhas dificuldades centraram-se na falta de experiência com a tecnologia utilizada, nomeadamente o facto de ser a primeira vez que implementei algo em VBA e VB6, uma linguagem que não suporta totalmente o paradigma de programação orientada a objectos e de ter de absorver novos conceitos como o registo de uma *dll* e o funcionamento do registo do *Windows*. Ainda do ponto de vista técnico, o contacto com o VBA foi uma novidade, não tinha muito conhecimento sobre as capacidades da linguagem e a forma como a equipa de desenvolvimento do departamento explora esta tecnologia integrando funcionalidades, que de outra forma

sacrificariam o conceito de aplicação *standard*, através de folhas *Microsoft Excel*, com programação nos bastidores, completamente integradas na aplicação.

Existe também a componente social deste projecto, na medida em que este exigiu a implementação no cliente e essa era, à partida, uma situação para a qual eu não estava preparado. Ao longo do percurso académico existem sempre os chavões relativos à falta de compreensão entre o cliente e o programador; na minha opinião, nunca é demais reforçar essa ideia. Ganhei a noção que no campo da relação com clientes toda a experiência é bem-vinda, porque os requisitos levantados hoje junto do cliente, amanhã já são totalmente diferentes. É necessário um esforço extra quer da parte de quem desenvolve novas tecnologias, quer de quem tem o conhecimento do negócio para que se chegue a um ponto consensual. Acho que ganhei bastante em ter de efectuar deslocações ao cliente; embora no inicio olhasse de forma relutante para essas situações, ganhei consciência de que não existe melhor forma de perceber o negócio do que através do contacto directo com quem lida, por vezes uma vida inteira, com a actividade comercial.

O projecto *knowledge base* teve um objectivo totalmente diferente do anterior, o que me permitiu obter conhecimento noutras áreas. Sendo um projecto cujo destinatário é a população do departamento e cuja função principal é aperfeiçoar processos dentro do mesmo, permitiu-me tomar conhecimento da funcionalidade dentro do departamento ao nível da resolução de problemas com as aplicações, ou o processo de testes de *software* do departamento.

Como já foi referido neste documento, existem duas equipas distintas no departamento, a equipa de desenvolvimento que é responsável pela implementação e manutenção das aplicações e a equipa de suporte que é responsável pela assistência técnica aos clientes. O projecto teve o objectivo de melhorar a ponte de ligação entre as duas equipas. Esta ponte permite que a equipa de suporte técnico reporte ocorrências dos clientes para a equipa de desenvolvimento poder fazer a sua avaliação e respectiva correcção. O grande problema do processo actual reside no facto de não existir um método formal da passagem de informação entre as duas equipas; assim sendo, não é difícil imaginar o mesmo problema ser reportado várias vezes. Outro problema é o facto de não existir um local onde a equipa de suporte possa verificar se já existe uma solução para o problema.

As falhas detectadas são colmatadas pelo repositório de conhecimento, que permite centralizar a informação disponível permitindo aos membros da equipa de suporte

técnico terem uma fonte de informação sempre disponível e permite aos membros da equipa de desenvolvimento não terem de resolver o mesmo problema vezes sem conta.

O projecto SigCRM foi desenvolvido em moldes diferentes dos anteriores, a começar por ser um projecto de equipa e também devido à sua dimensão muito superior. O trabalho desenvolvido neste projecto passou por desenvolver as classes base para que se desenvolvesse sobre elas.

Por ser a primeira vez que me integrei num projecto de grande dimensão, senti por vezes, alguma dificuldade em visualizar o âmbito geral do mesmo; contudo, por retratar uma área relativamente simples de compreender, era mais fácil de retomar o rumo certo. Quanto ao trabalho em equipa sinto que, a longo prazo, correu tudo como esperado e é relevante o facto de ter sido posta em acção uma metodologia de desenvolvimento muito prática que permitiu manter a equipa focada no seu trabalho e ao mesmo tempo atenta às dificuldades dos restantes elementos.

Tecnicamente foi um projecto com um grau de dificuldade maior, que permitiu ter conhecimento do nível de esforço necessário para a conclusão de projectos desta dimensão. Permitiu-me ter contacto com novas técnicas de programação, em concreto, Ajax. Embora tenhamos optado por recorrer a uma biblioteca existente (Anthem), ao invés de "inventar a roda", obrigou-me a estudar os pormenores da tecnologia.

A um nível mais global julgo que a equipa de desenvolvimento tinha a ganhar se existisse uma aposta mais forte numa equipa de testes; o processo actual contém alguns pontos de fragilidade como, por exemplo ser a equipa de suporte técnico a fazê-los e muitas vezes ser o próprio programador a efectuar os testes. O sector da documentação também não é muito privilegiado; na minha opinião o departamento tinha muito a ganhar com a criação de documentação técnica de apoio ao desenvolvimento, como diagramas de casos de uso ou diagramas de fluxo de dados, na medida em que simplificam o processo de decomposição de uma tarefa. Contudo, a falta de documentação é compensada com o nível de disponibilidade que os elementos mais seniores têm para facultar a sua visão abrangente do negócio aos elementos mais juniores.

Esta, que foi a minha primeira experiência no mundo do trabalho, foi extremamente positiva, tanto pelo sentido de responsabilidade que foi necessário desenvolver, como pelo sentimento de confiança que sempre foi depositado no meu trabalho.

4.2 Trabalho futuro

O projecto *knowledge base* poderá ser melhorado através da ligação ao *Microsoft Project Server*, aplicação que o departamento já utiliza para gerir os projectos. Desta forma teríamos um repositório ainda mais centralizado, em que um membro da equipa de desenvolvimento poderia identificar as suas tarefas, geri-las e reportar directamente para os membros da equipa de suporte a sua conclusão.

No projecto da facturação electrónica julgo que seria um passo importante desenvolver a aplicação de comunicação com o serviço de repositório de documentos electrónicos. Deste modo o módulo poderia ser directamente integrado na SigGC com um sistema de verificação de correio de documentos electrónicos.

A aplicação SigCRM carece, na minha opinião, de personalização ao nível do aspecto gráfico. Seria interessante cada cliente poder ter o aspecto da aplicação de acordo com a sua imagem.

Lista de acrónimos

Ajax	Asynchronous JavaScript and XML
ASP	Active Server Pages
COM	Component Object Model
CRM	Customer Relationship Management
DAL	Data Access Layer
DLL	Dynamic Linked Library
ERP	Enterprise Resource Planning
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IIS	Internet Information Services
LOV	List of Values
PHP	Hypertext Preprocessor
SIG	Sistema Integrado de Gestão
SigAV360	Avaliação de Desempenho 360º
SigCRM	Aplicação CRM do ERP Sig
SigCTB	Aplicação de Contabilidade e Gestão Financeira do ERP Sig
SigGC	Aplicação de Gestão Comercial do ERP Sig
SigGP	Aplicação de Gestão Pessoal do ERP Sig
SQL	Structured Query Language
TI	Tecnologias de Informação
URL	Uniform Resource Locator
VB6	Visual Basic 6.0
VBA	Visual Basic for Applications
XML	Extensible Markup Language
XSD	XML Schema

Bibliografia

[1] – Mike Murach. ASP.NET 2.0 web programming with C# 2005. Mike Murach & Associates, Inc., 2005

[2] – *MSDN training, Developing Microsoft ASP.Net Applications using Visual Studio .Net* (http://www.microsoft.com/learning/syllabi/en-us/2310Bfinal.aspx)

[3] – Visual Basic design patterns : VB6 / VB.NET. James W. Cooper, Addison-Wesley, 2001